INTRODUCTION TO JCL OBJECTIVES

---

1. WHAT IS JCL?

1.1 JOB CONTROL LANGUAGE consists of control statements that:

- introduce a computer job to the operating system
- request hardware devices
- direct the operating system on what is to be done in terms of running applications and scheduling resources

JCL is not used to write computer programs. Instead it is most concerned with input/output--- telling the operating system everything it needs to know about the input/output requirements. It provides the means of communicating between an application program and the operating system and computer hardware.

1.2 IS JCL DIFFICULT? ... NOT NECESSARILY!

The role of JCL sounds complex and it is---JCL can be downright difficult.

JCL can be difficult because of the way it is used. A normal programming language, however difficult, soon becomes familiar through constant usage. This contrasts with JCL in which language features are used so infrequently that many never become familiar.

JCL can be difficult because of its design - JCL:

- consists of individual parameters, each of which has an effect that may take pages to describe
- has few defaults--must be told exactly what to do
- requires specific placement of commas and blanks
- is very unforgiving--one error may prevent execution

JCL is not necessarily difficult because most users only use a small set of similar JCL that never changes from job to job.

1.3 HOW DO YOU SEND INFORMATION TO THE COMPUTER?

BATCH PROCESSING VS. INTERACTIVE PROCESSING

Interactive Processing means that you give the computer a command and the computer responds to your command request. It is more like a conversation.

Batch Processing means that you give the computer a whole group of commands, usually in the form of some sort of program you have written, and have the computer process this group of commands. It is more like writing a letter.

2. BASIC SYNTAX OF JCL STATEMENTS

//NAME	OPERATION	OPERAND,OPERAND,OPERAND	COMMENTS
|	|	|	|
name field	operation field	operand field	comment field

name field - identifies the statement so that other statements or the system can refer to it. The name field must begin immediately after the second slash. It can range from 1 to 8 characters in length, and can contain any alphanumeric (A to Z) or national (@ $ #) characters.
operation field - specifies the type of statement: JOB, EXEC, DD, or an operand command.
operand field - contains parameters separated by commas. Parameters are composites of prescribed words (keywords) and variables for which information must be substituted.
comments field - optional. Comments can be extended through column 80, and can only be coded if there is an operand field.

General JCL Rules:

- Must begin with // (except for the /* statement) in columns 1 and 2
- Is case-sensitive (lower-case is just not permitted)
- NAME field is optional
- must begin in column 3 if used
- must code one or more blanks if omitted
- OPERATION field must begin on or before column 16
- OPERATION field stands alone
- OPERANDS must end before column 72
- OPERANDS are separated by commas
- All fields, except for the operands, must be separated by one blank.

2.1 CONTINUATION OF JCL STATEMENTS

```
//LABEL  OPERATION  OPERAND, OPERAND,
//    OPERAND,OPERAND,
//   OPERAND,
//      OPERAND
```

When the total length of the fields on a control statement exceeds 71 columns, continue the fields onto one or more following statements.

- Interrupt the field after a complete operand (including the comma that follows it) at or before column 71
- Code // in columns 1 and 2 of the following line
- Continue the interrupted statement beginning anywhere in columns 4 to 16.

2.2 COMMENTING JCL

```
//* THIS IS A COMMENT LINE
```

JCL should be commented as you would any programming language. The comments statement contains //* in columns 1 to 3, with the remaining columns containing any desired comments. They can be placed before or after any JCL statements following the JOB statement to help document the JCL. Comments can also be coded on any JCL statement by leaving a blank field after the operand field.

3. THREE TYPES OF JCL STATEMENTS

JOB Identifies the beginning of a job

EXEC Indicates what work is to be done

DD Data Definition, i.e., Identifies what resources are needed and where to find them

4. THE JOB STATEMENT

The JOB statement informs the operating system of the start of a job, gives the necessary accounting information, and supplies run parameters. Each job must begin with a single JOB statement.//jobname JOB USER=userid

jobname - a descriptive name assigned to the job by the user which is the banner on your printout
- any name from 1 to 8 alphanumeric (A-Z,0-9) or national ($,@,#) characters
- first character must be alphabetic or national

JOB - indicates the beginning of a job

Userid - a 1 to 7 character user identification assigned to access the system

4.1 ADDITIONAL OPERANDS OF THE JOB STATEMENT

//jobname JOB USER=userid, TIME=m, MSGCLASS=class, NOTIFY=userid

USER=userid Identifies to the system the user executing the job
TIME=m Total machine (m)inutes allowed for a job to execute
MSGCLASS=class Output class for the job log
NOTIFY=userid User to receive a TSO message upon completion of a job

4.1.1 MSGCLASS

The MSGCLASS parameter allows you to specify the output class to which the operating system MVS is to write the job log or job entry subsystem (JES) messages. If you do not code the MSGCLASS parameter, MSGCLASS=J is the default and will be used. MSGCLASS=J indicates the output will be printed on 8 1/2" by 11" hole paper.

This includes the following:

1. Job Entry Subsystem (JES) Messages
2. Error Messages
3. JCL Statements
4. Dataset Dispositions
5. Accounting information

Here is a list of available output classes:

- A greenbar paper
- 5-9 TSO held output
- C-Z IBM page printer output classes. See chart in section "Sys-out Classes for IBM Print" on page 28.

5. THE EXEC STATEMENT

Use the EXEC (execute) statement to identify the application program or cataloged or in-stream procedure that this job is to execute and to tell the system how to process the job.

```
//stepname EXEC procedure,REGION=####K
```

or

```
//stepname EXEC PGM=program,REGION=####K
```

stepname - an optional 1 to 8 character word used to identify the step
EXEC - indicates that you want to invoke a program or cataloged procedure
procedure - name of the cataloged procedure to be executed
program - name of the program to be executed
REGION=####K - amount of storage to allocate to the job

5.1 PROGRAMS AND CATALOGED PROCEDURES

```
// EXEC PGM=pgmname
```

A program referred to on the EXEC PGM= statement is a compiled and linked version of a set of source language statements that are ready to be executed to perform a designed task. It is also known as an executable load module. It must reside in a partitioned dataset.

```
// EXEC cataloged-procedure-name
```

Because the same set of JCL statements are often used repeatedly with little or no change they can be stored in cataloged procedures. JCL provides programmers with the option of coding these statements only once, recording and cataloging the statements under an appropriate name in a procedure library, and then invoking these statements through an EXEC statement. Such a previously established set of JCL statements is known as a "cataloged procedure." The effect of using a cataloged procedure is the same as if the JCL statements in the procedure appeared directly in the input stream in place of the EXEC statement calling the procedure. This saves the user from writing lengthy error-prone JCL statements. In short, this is JCL that does not have to be included in batch jobs.

* Note that within cataloged procedures a program will be executed.

5.1.1 Modifying Cataloged Procedures

Additional statements may be added to the JCL comprising a cataloged procedure at the time of invocation. Also, operand values on existing JCL statements may be altered or parameters defined in the procedure may be substituted at invocation time.

The value in the override statement replaces the value for the same parameter in the cataloged procedure. Cataloged procedure statements must be overridden in the same order as they appear in the procedure.

```
//procstepname.ddname DD parameter=value
```

You can modify cataloged procedures by:

- Overriding parameters in an existing DD statement
- Adding DD statements to the procedure. To add new DD statements let them follow any changed DD cards for that step.

To modify an existing DD statement, only those operands to be changed need be coded on the modifying DD statement. The remaining operands of the DD statement within the procedure will be unchanged. If more than one DD statement in a procedure is to be modified, the modifying DD statements must be placed in the same order as the original DD statements occur in the procedure.

To add a DD statement to an existing procedure, place the DD statement after the procdure invocation EXEC statement and any modified DD statements within the job step.

```
// EXEC SAS
//NEWDD DD DSN=user.file,DISP=SHR
```

When looking at your output, the following symbols will determine what kind of statement is indicated when your job runs:

// Indicates JCL statements

XX Indicates cataloged procedure statements

X/ Indicates a modified cataloged procedures statement

6. DATA DEFINITION (DD) STATEMENT

A DD (Data Definition) statement must be included after the EXEC statement for each data set used in the step. The DD statement gives the data set name, I/O unit, perhaps a specific volume to use, and the data set disposition. The system ensures that requested I/O devices can be allocated to the job before execution is allowed to begin.

The DD statement may also give the system various information about the data set: its organization, record length, blocking, and so on.

```
//ddname DD operand,operand,etc.
```

ddname - a 1 to 8 character name given to the DD statement

DD - DD statement identifier

operand - parameters used to define the input or output dataset

The DD Statement

- appears after an EXEC statement
- gives the system information on many things, including the dataset attributes, the disposition of the dataset when the job completes, and which input/output device(s) to use

6.1 DD STATEMENT FOR INSTREAM DATA

Instream data is perhaps the most common form of input. To include data in the input stream, code:

```
//ddname DD *
.
.
.
/* (to specify end of data)
```

SYSIN is often used as a ddname for instream data. The /* marks the end of the data.

6.2 EXAMPLE PROGRAMS WITH INSTREAM DD STATEMENTS

6.2.1 Fortran Example

In this example, the userid UGUSER is executing a Fortran procedure with the Fortran program inserted after the FORT.SYSIN line and the data inserted after the GO.SYSIN line.

```
//FORTRUN JOB USER=UGUSER
// EXEC FORTVCLG,REGION=1500K
//FORT.SYSIN DD *
FORTRAN statements
/*
//GO.SYSIN DD *
data lines
/*
//
```

6.2.2 SAS Example

In this example, the userid UGIBM is executing a SAS procedure with the SAS program inserted after the SYSIN line.

```
//SASRUN JOB USER=UGIBM
// EXEC SAS,REGION=1500K
//SYSIN DD *
SAS statements
/*
//
```

6.3 DATA DEFINITION (DD) STATEMENT FOR DISK DATASETS

```
//ddname DD UNIT=unittype,
//            DSN=userid.name,
//            DISP=(beginning,normal-end,abnormal-end),
//            SPACE=(TRK,(primary,secondary,directory)),
//            RECFM=xx,LRECL=yy,MGMTCLAS=retainx
```

ddname - data definition name; a 1-8 character word of your choice, must begin with a letter or $, @, #

DD - DD statement identifier

UNIT = unittype - an I/O unit is a particular type of I/O device: a disk, tape, etc. UNIT=SYSDA refers to the next available disk storage device.

DSN=userid.name DSN parameter names the data set. Data sets can be temporary or nontemporary. A temporary data set is created and deleted within the job, whereas nontemporary data sets can be retained after the job completes. A data set name can contain up to 44 characters including periods.

Ex. UGIBM.DATA

mgmtclas - MGMTCLAS specifies the name of the Management Class which is a set of specifications for the way the storage occupied by the data set should be treated by SMS. Generally, this deals with how long you want to keep this data set around. UCNS has set up the following management classes:

MGMTCLAS	Days Retention	MGMTCLAS	Days Retention
RETAIN0	0 (DEFAULT)	RETAIN8	8
RETAIN1	1	RETAIN9	9
RETAIN2	2	RETAIN10	10
RETAIN3	3	RETAIN14	14
RETAIN4	4	RETAIN28	28
RETAIN5	5	RETAIN56	56
RETAIN6	6	RETAIN95	95
RETAIN7	7	STANDARD	(18 months past last use)

6.4 DATA DEFINITION (DD) STATEMENT FOR TAPE DATASETS

```
//ddname DD UNIT=unittype,VOL=SER=unitname,
//              DSN=filename,
//              DISP=(beginning,normal-end,abnormal-end),
//              DCB=(RECFM=xx,LRECL=yy,BLKSIZE=zz,DEN=density),
//              LABEL=(file#,labeltype,,mode)
```

ddname - data definition name; a 1-8 character word of your choice, must begin with a letter or $, @, #

DD - DD statement identifier

UNIT=unittype - an I/O unit is a particular type of I/O device: disk, tape, etc.
TAPE16 refers to a 1600 bpi tape drive
TAPE62 refers to a 6250 bpi tape drive
TAPECA refers to a 38K or XF catridge tape drive

VOL=SER=unitname - VOL=SER parameter is needed if the data set is to be placed on a specific tape volume. This refers to the volume serial number on an internal tape label.

DSN=filename - DSN parameter names the file on the tape. The filename can be from 1 to 17 characters in length.
Ex. COWDATA

6.4.1 Disposition (DISP) Parameters

The DISP parameter describes the current status of the data set (old, new, or to be modified) and directs the system on the disposition of the dataset (pass, keep, catalog, uncatalog, or delete) either at the end of the step or if the step abnormally terminates. DISP is always required unless the data set is created and deleted in the same step.

// DISP = (beginning, normal-termination ,abnormal-termination)

```
            __________    ___________________   ___________________
               |               |                   |
             NEW           CATLG                DELETE
             OLD            KEEP                 KEEP
             SHR            PASS                CATLG
             MOD           DELETE              UNCATLG
                          UNCATLG
```

6.4.1.1 Beginning Dispositions

This is the status of the data set at the beginning of the step. If the data set is new, the system creates a data set label; if it is old, the system locates it and reads its label

NEW creates a new data set

OLD designates an existing data set; it can be an input data set or an output data set to rewrite

SHR identical to OLD except that several jobs may read from the data set at the same time.

MOD modifies a sequential data set - positions the pointer at the end of the data set in order to add new data to the data set.

6.4.1.2 Normal Termination and Abnormal Termination Dispositions

Normal disposition, the second term in the DISP parameter, indicates the disposition of the data set when the data set is closed or when the job terminates normally. The abnormal dispositions, effective only if the step abnormally terminates, are the same as normal dispositions except that PASS is not allowed.

PASS passes the data set on to subsequent job steps, and each step can use the data set once.

KEEP keeps nontemporary data sets.

DELETE deletes data sets.

CATLG catalogs a nontemporary data set. CATLG is similar to KEEP except that the unit and volume of the data set are recorded in the catalog along with the data set name.

UNCATLG uncatalogs a data set. UNCATLG is the same as KEEP except that the data set name is removed from the catalog.

6.4.1.3 Examples of DISP parameters

// DISP=SHR	read from
// DISP=OLD	write to
// DISP=(NEW,CATLG,DELETE)	create and catalog; delete if there is a system abend
// DISP=(OLD,PASS)	dataset already exists; pass it to the next step
// DISP=MOD	write to the bottom of an existing dataset

6.4.2 SPACE Parameter

All new data sets on disk volumes must be allocated space. Storage on disk volumes can be allocated in units of blocks, cylinders, tracks, kilobytes, and bytes.

The space may be requested as a primary and a secondary amount. The primary amount is allocated when the data set is opened with a disposition of NEW. The secondary amount is allocated if the primary amount is exceeded.

The primary amount can be conservative, with the secondary amount providing a reserve. The secondary amount provides for data set growth over time.

// SPACE=(TRK,(primary,secondary,directory))

primary receive this amount of space initially

secondary receive this amount of space each time more is needed (up to 15 times)

directory reserve this amount of blocks to keep the directory of a partitioned dataset (NOT USED for a sequential dataset)
1 directory block allows for 5 members in a partitioned dataset

Total Space = (1 * primary) + (15 * secondary)

6.4.2.1 Partitioned Dataset vs. Sequential Dataset

Any named collection of data is called a data set. A partitioned dataset consists of multiple files within one data structure. A sequential dataset consists of one file within a data structure.

Partitioned Dataset

- individual members are read/manipulated without disturbing other members
- it is advisable to never write directly to a partitioned dataset in your program
- on DD statememt: DSN=userid.file(member)
- call from editor: file(member)

Sequential Dataset

- dataset must be read from top to bottom
- on DD statement: DSN=userid.file
- call from editor: file

```
     PARTITIONED DATASET              SEQUENTIAL DATASET
     DSN='userid.data-set-         DSN='userid.data-set-name'
        name(member)'

      _______________               _______________
     |  DIRECTORY  |               |               |
      |-----------------|           |               |
     |   MEMBER1   |               |               |
      |-----------------|          |    DATASET    |
     |   MEMBER2   |               |               |
      |-----------------|           |               |
     |     .......     |           |               |
      |-----------------|           |               |
     |   MEMBERn   |               |               |
    |-------------------------|     |_______________|
     |_______________|
```

A partitioned dataset differs from a sequential dataset in that it has a directory of its members. Whenever you refer to a member of a partitioned dataset, you include the member name in parentheses.

6.4.2.2 DFSMS (System Managed Datasets)

On the TSO service, data sets are typically created and reside on disk volumes. A volume is a standard unit of storage. These disk volumes are referred to as DASD, which stands for Direct Access Storage Device. Each block of data on a DASD volume has a distinct location and a unique address, making it possible to find any record without extensive searching. One DASD volume can be used for many different data sets, and space on it can be reallocated and reused.

At the University of Georgia, you are now required to utilize the Data Facilities Storage Management Subsystem (DFSMS) to establish permanent data sets. The Storage Management Subsystem (SMS) is an operating environment that automates the management of storage. With SMS, users can allocate data sets more easily. The data sets allocated through the Storage Management Subsystem are called system-managed. System-managed means that the system determines data placement and automatically manages data availability, performance, space, reclamation, and security. One of the most beneficial goals of System-managed storage is to relieve users of performance, availability, space, and device management details.

DFSMS stores data in a device-independent format so that it can easily move the data to any of the following devices:

- 3490E Magnetic Catridge Tape
- DASD for models 3380 and 3390

The migration and movement of data depends on such factors as:

- Management Class
- Data set Usage
- Minimum percent free space on a DASD Volume
- Request by storage administrator or user

DFSMS records the location of each dataset it moves in a control data set. The actual migration is handled by DFHSM (Data Facilities Hierarchical Storage Manager). DFHSM is a DASD management product tool for managing low-activity and inactive data.

Data sets that have reached the end of their retention period (expired) will be deleted. Data sets with a management class of STANDARD will be deleted if the data set has not been referenced for a period of eighteen months. A notification will be sent to the user after a STANDARD data set has not been referenced for six months informing the user of the STANDARD deletion policy. At this time the data set will be moved to tape.

All data sets that have a management class of RETAIN95 or STANDARD will be automatically backed up by DFHSM. Two copies of each will be kept. The change indicator will trigger the backup after the first backup is made. A user can use the HBACK command to add non-STANDARD and non-RETAIN95 data sets to this.

6.4.2.3 SPACE Parameter (Partitioned Dataset)

// SPACE=(TRK,(primary,secondary,directory))

This SPACE example allows a total of 40 tracks for the dataset with 1 block of space reserved for the directory.

// SPACE=(TRK,(10,2,1))

Total Space = (1*10) + (15*2) = 10 + 30 = 40 tracks
Directory = (1*5) = 5 members/dataset

This SPACE example allows a total of 100 tracks for the dataset with 8 blocks reserved for the directory.

// SPACE=(TRK,(25,5,8))

Total Space = (1*25) + (15*5) = 25 + 75 = 100 tracks
Directory = (8*5) = 40 members/dataset
* Note that the track capacity for 3380 is 1 track = 47,476 characters.

6.4.2.4 SPACE Parameter (Sequential Dataset)

// SPACE=(TRK,(primary,secondary))

This SPACE example allows a total of 53 tracks for the dataset.

```
// SPACE=(TRK,(8,3))
```

Total Space = (1*8) + (15*3) = 8 + 43 = 53 tracks

This SPACE example allows a total of 520000 bytes for the dataset.

```
// SPACE=(80,(5000,100))
```

Total Space = (1*400000) + (15*8000) = 400000 + 120000 = 520000 bytes
* Note that directory blocks are always 0 for sequential datasets; therefore, the directory parameter is NOT USED for sequential datasets.

6.4.3 Data Set Attributes

With SMS, you do not need to use the DCB parameter to specify data set attributes. ALL of the DCB keyword subparameters (record length, record format, and blocksize) can be specified without the need to code DCB=.

For example, the following DD statement:

```
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000)
```

can be specified as:

```
// RECFM=FB,LRECL=80
```

The blocksize parameter can be omitted because SMS will select the optimum blocksize.

RECFM=xx specifies the record format. The format can be one or more of the following characters:

F fixed-length records
V variable-length records
U undefined-length records
FB fixed and blocked
FBA fixed, blocked, with ANSI carriage control characters
VB variable and blocked
VBA variable, blocked, with ANSI carriage control characters
LRECL=yy specifies the length of records

equal to the record length for fixed-length records

equal to the size of the largest record plus the 4 bytes describing the record's size for variable-length records

omit the LRECL for undefined records

LRECL can range from 1 to 32760 bytes

BLKSIZE=zz specifies the blocksize if you wish to block records

- must be a multiple of LRECL for fixed-length records

- must be equal to or greater than LRECL for variable-length records
- must be as large as the longest block for undefined-length records
- BLKSIZE can range from 1 to 32760 bytes

6.4.3.1 Fixed Block

```
// RECFM=FB,LRECL=80,BLKSIZE=9040
```

This dataset will have fixed length records with a length of 80. There will be 113 records of data per block.

BLKSIZE/LRECL = 9040/80 = 113 records of data per block

.4.3.2 Variable Block

```
// RECFM=VB,LRECL=255,BLKSIZE=3120
```

This dataset will have variable length records with a maximum of 255 characters. The blocksize of the dataset will be 3120.

6.4.4 DCB and LABEL Parameters for Tapes

RECFM=xx specifies the record format. The format can be one or more of the following characters:
F fixed-length records
V variable-length records
U undefined-length records
FB fixed and blocked
FBA fixed, blocked, with ANSI carriage control characters
VB variable and blocked
VBA variable, blocked, with ANSI carriage control characters
LRECL=yy specifies the length of records
equal to the record length for fixed-length records
equal to the size of the largest record plus the 4 bytes describing the record's size for variable-length records
omit the LRECL for undefined records
LRECL can range from 1 to 32760 bytes

BLKSIZE=zz specifies the blocksize if you wish to block records
must be a multiple of LRECL for fixed-length records
must be equal to or greater than LRECL for variable-length records
must be as large as the longest block for undefined-length records
BLKSIZE can range from 1 to 32760 bytes

DEN=density measures the number of bits that are stored in a unit of measurement on the tape. This measurement is commonly referred to as BPI (bits per inch).

densities are represented in the DEN parameter as follows:

2	800 BPI
3	1600 BPI
4	6250 BPI
*	38K OR XF BPI

when adding files to an existing tape all new files will be written at the same density as the first file no matter if you specify differently.

The LABEL parameter tells the type of label, the relative file number on the tape, and whether the data set is to be protected for input or output.

// LABEL=(file#,labeltype,,mode)

file - the relative file number on the tape (1-4 digits)

type - the type of label on the tape
NL No Label
SL Standard Label
AL American National Standard Label
BLP Bypass Label Processing

mode IN/OUT parameter
IN protects the file from being opened for output
OUT protects it from being opened for input

// LABEL=(3,SL,,IN) File 3 on a SL tape can only be read

// LABEL=(1,NL,,OUT) File 1 on a NL tape is open for output

6.5 EXAMPLE PROGRAMS WITH DD STATEMENTS

6.5.1 Fortran Job for Reading from Disk Dataset

In this example, the userid UGIBM is executing a Fortran procedure with the Fortran program inserted after the FORT.SYSIN line, and the data is being read by UNIT 3 from the data set UGIBM.FOOD.

```
//FORTRUN JOB USER=UGIBM
//     EXEC FORTVCLG,REGION=2000K
//FORT.SYSIN DD *


      read(3,10) x,y
   10 format(1x,f4.1,1x,f4.1)


/*
//GO.FT03F001 DD DSN=UGIBM.FOOD,UNIT=SYSDA,DISP=SHR
/*
//
```

6.5.2 SAS Job for Reading from Disk Dataset

In this example, the userid UGIBM is executing a SAS procedure with the SAS program inserted after the SYSIN line and the data is being read from the data set UGIBM.DATA.

```
//SASRUN JOB USER=UGIBM
//     EXEC SAS,REGION=2000K
//OLDDATA DD DSN=UGIBM.DATA,UNIT=SYSDA,DISP=SHR
//SYSIN DD *

data one; infile olddata;
input x y z;

/*
//
```

6.5.3 Fortran Job for Writing to Disk Dataset

In this example, the userid UGABC is executing a Fortran procedure with the Fortran program inserted after the FORT.SYSIN line, and the results are written by UNIT 8 to the data set UGABC.NEWFILE. This dataset will only be retained for 7 days. If the user decides to keep the dataset for a longer period of time, the ALTER command can be issued. For example, ALTER 'UGABC.NEWFILE' MGMTCLAS(RETAIN14), will keep the dataset around for another week. Since the BLKSIZE is not specified, SMS will determine the most efficient blocksize.

```
//FORTRUN JOB USER=UGABC
// EXEC FORTVCG,REGION=2000K
//FORT.SYSIN DD *

      write(8,10) x,y
10    format(1x,f4.1,1x,f4.1)

/*
//GO.FT08F001 DD DSN=UGABC.NEWFILE,UNIT=SYSDA,
// DISP=(NEW,CATLG,DELETE),
// SPACE=(TRK,(40,10),RLSE),
// RECFM=FB,LRECL=80,MGMTCLAS=RETAIN7
/*
//
```

6.5.4 SAS Job for Writing to Disk Dataset

In this example, the userid UGXYZ is executing a SAS procedure with the SAS program inserted after the SYSIN line and the results are written to the data set UGXYZ.SAS.DATA. This dataset, based on the management class of standard, will be retained on the system as long as the user utilizes it.

```
//SASRUN JOB USER=UGXYZ
// EXEC SAS,REGION=2000K
//NEWDATA DD UNIT=SYSDA,
// DSN=UGXYZ.SAS.DATA,DISP=(NEW,CATLG,DELETE),
// SPACE=(TRK,(40,10),RLSE),
// RECFM=FB,LRECL=80,MGMTCLAS=STANDARD
//SYSIN DD *

   data example;
   input x y z;
   cards;
   1 2 3
   4 5 6
   ;
   data _null_; set example; file newdata;
   put x y z;
   return;

/*
//
```

7. DD STATEMENT FOR PRINTED OUTPUT

The SYSOUT parameter provides a convenient means of routing output to printers or other devices.

//ddname DD SYSOUT=class

ddname - a 1 to 8 character name given to the DD statement

DD - DD statement identifier

SYSOUT=class - defines this data set as a system output data set, usually called a sysout data set and assigns this sysout data set to an output class. Here is a list of available output classes:

- A greenbar paper
- 5-9 TSO held output
- C-Z IBM page printer output classes. See chart in section "Sysout Classes for IBM Print" on page 28.

* same class as specified on the MSGCLASS parameter

* Note that if a sysout data set has the same class as the MSGCLASS parameter, the job log appears on the same output listing as this sysout data set.

7.1 SYSOUT CLASSES FOR IBM PRINT

SYSOUT* CLASS	PAPER	ORIENTATION	SIDES	IMAGES	CHAR/LINE	LINES/IMAGE
C	hole	landscape	1	1	132	62
D	hole	portrait	1	1	80	62
F	bond	landscape	2	1	132	62
G	bond	portrait	2	1	80	62
H	bond	landscape	2	2	90	80
I	bond	portrait	2	1	80	70
J	hole	landscape	2	1	132	62
K	hole	portrait	2	1	80	62
Q	bond	landscape	1	1	132	62
S	bond	portrait	1	1	80	62
W	hole	portrait	2	2	132	62
X	bond	portrait	2	2	132	62
Y	bond	landscape	2	1	132	62
Z	hole	landscape	2	1	132	62

Graybar Overlay: The paper used for classes C,F,H,J, and Q is shaded to mimic standard greenbar paper.

Service Site Page Printers: To route your output to one of the page printers in the Journalism or Aderhold sites, you must use a SYSOUT class or message class (MSGCLASS) with a paper type of "hole" (C, D, J, K, W, or Z).

Line Printer SYSOUT Class: To route your output to a line printer, use a SYSOUT class or message class (MSGCLASS) of A. Your output will be printed on greenbar paper.

Special Hold Classes: To route your output to a hold queue, use a SYSOUT class or message class (MSGCLASS) of 5, 6, 7, 8, or 9.
----

* SYSOUT class values may be used for message class (MSGCLASS).

8. OUTPUT JCL STATEMENT

The OUTPUT statement is used to specify processing options for a system output data set. These options are used only when the OUTPUT statement is explicitly or implicitly referenced by a sysout DD statement.

//name OUTPUT FORMDEF=fdef,PAGEDEF=pdef,CHARS=ch,FORMS=form,
// COPIES=n,DEST=dest,DEFAULT=dd

name - a 1 to 8 character name given to the OUTPUT statement

OUTPUT - OUTPUT statement identifier

FORMDEF=fdef - specifies whether to use duplex or simplex and overlay

DUP1	duplex, no overlay
DUP2	duplex, grey bar overlay
SMP1	simplex, no overlay
SMP2	simplex, grey bar overlay

PAGEDEF=pdef - specifies the logical page length and width, fonts, lines within a page, and multiple logical pages on a physical page

POR1	portrait, 62 lines at 6 lines per inch
POR2	portrait, 70 lines at 6 lines per inch
POR3	portrait, 2 up format, 62 lines per frame
LAN1	landscape, 62 lines at 8 lines per inch
LAN2	landscape, 2 up format, 80 lines per frame

CHARS=ch - names the character set

GT12	Gothic font at 12 characters per inch
GT13	Gothic font at 13 characters per inch
GT18	Gothic font at 18 characters per inch

FORMS=form - specifies the type of form

BOND	plain bond paper
HOLE	3 hole drilled bond paper

COPIES=n - specifies how many copies of the sysout data set are to be printed

DEST=dest - specifies a printer destination for the sysout data set

LOCAL	Boyd Graduate Studies
SSS02	Journalism Building
SSS03	Aderhold Building
NCT19	Brooks Hall

DEFAULT=dd - specifies that this OUTPUT statement can or cannot be referenced by a sysout DD statement

YES	specifies that this is the default OUTPUT statement for for all print files within a job
NO	specifies that this is not the default OUTPUT statement

* Note if you are going to HOLD a print dataset with an OUTPUT statement associated with it, you will have to fully specify all of the parameters on the OUTPUT statement. Contact the UCNS Helpdesk for more details.

8.1 EXAMPLES OF THE OUTPUT STATEMENT

To request that the "gray bar" overlay not print on a landscape sysout class for all print files, code:

```
//OUT1 OUTPUT FORMDEF=DUP1,DEFAULT=YES
```

To request simplex (one sided) printing for a duplex sysout class for all print files, code:

```
//OUT1 OUTPUT FORMDEF=SMP1,DEFAULT=YES
```

To request multiple copies of a print file, code:

```
//OUT1 OUTPUT COPIES=12
```

- with -

```
//ddname DD SYSOUT=F,OUTPUT=*.OUT1
```

9. JES3 CONTROL STATEMENTS

9.1 //*MAIN STATEMENT

The //*MAIN statement is used to define the processor requirements for the current job. It specifies what time the job will be executed, how many lines in the job, and where the job is to be printed.

//*MAIN CLASS=x,LINES=y,ORG=UGAIBM1.org

CLASS=x specifies the job class for this job

B	Batch (default) Anytime daily
NITE	6:00 PM until 7:00 AM daily
WEEKEND	6:00 PM Friday until 7:00 AM Monday
Z	1:30 AM until 7:00 AM daily, weekends, holidays
BL	6:00 PM until 7:00 AM daily, more than 8 MEG region
ZL	1:30 AM until 7:00 AM daily, more than 8 MEG region

LINES specifies the maximum number of lines of data to be printed from this job in multiples of a thousand (default = 5000)

ORG=UGAIBM1.org specifies a printer destination for the sysout dataset

LOCAL	Boyd Graduate Studies
SSS02	Journalism Building
SSS03	Aderhold Building
NCT19	Brooks Hall

9.2 JOB SCHEDULING SPECIFICATIONS

Since TSO is a time sharing operating system, it allows many people to use the computer at the same time in such a way that each is unaware that the computer is being used by others. Time sharing attempts to to maximize an individuals use of the computer, not the efficiency of the computer itself. In order to do this, job scheduling is used to assign jobs to a certain class in order to maximize the resources available to each user.

```
                    Priority
                      -------
             __________
BATCH  |               4
       |
       | __________   3
                 ____
       NITE    | ____   2
```

Any job can be scheduled to run at night or on the weekend by coding:

//*MAIN CLASS=NITE

or

//*MAIN CLASS=WEEKEND

9.2.1 Table of Resource Limitations for Job Scheduling

Priority	CPU Time (seconds)	Region (K)	Estimated Lines	Setups Required
6	0-30	0-2048	0-5000	0
4	32-120	2049-3072	5001-10000	0
3	121-300	3073-4096	10001-40000	1-3
2	301+	4097+	40000+	4+

9.3 //*OPERATOR STATEMENT

The //*OPERATOR statement is primarily used to issue a message to the operator requesting that the tape with the specific VRN, VSN and KEYWORD is to be mounted for the job.

//*OPERATOR VRN=#9999 VSN=U9999 KEY=HELP

VRN will be assigned when the tape is checked in. This identifies the tape in the tape library. Cartridge Tapes must have #C in the first two column positions on the vrn parameter.

VSN the volume serial number on the the internal label of a standard labeled tape. This must be the actual internal label. For non-labeled tapes, this can be any arbitrary name. In both cases, the VSN must match the VOL=SER parameter on the DD card.

KEY is a password that you will assign to the tape for security purposes. The operator will check the keyword in your JCL against the keyword on the tape before mounting the tape.

The //*Operator statement in the JCL stream is placed after the JOB or //*MAIN statement and before the EXEC statement. Each //*Operator statement should be referenced by a DD statement to read or write a file to the tape.

The following example illustrates the operator card with the referencing DD statement. Note that the VSN and the VOL=SER must be the same. In this example, the user is going to read the first file on a standard labeled (SL) tape with the name COWDATA. The tape is going to be read on a 6250 BPI tape drive.

```
//SEMINAR JOB USER=USERID
//*MAIN LINES=20
//*OPERATOR VRN=#1111 VSN=0211ZZ KEY=KFJD
//STEP1 EXEC ...
//INFILE     DD     UNIT=TAPE62,VOL=SER=0211ZZ,
//                  LABEL=(1,SL,,IN),
//                  DSN=COWDATA,
//                  DISP=(OLD,PASS)
```

10. EXAMPLE FORTRAN AND SAS PROGRAMS

10.1 EXAMPLE FORTRAN PROGRAM

In this example, the userid UGA001 is executing a Fortran procedure with the Fortran program inserted after the FORT.SYSIN line and the data inserted after the GO.SYSIN line. This job will not begin executing until after 6:00PM, and the output will be held to the terminal by the MSGCLASS=6.

```
//CONVERT JOB USER=UGA001,MSGCLASS=6,NOTIFY=UGA001
//*MAIN CLASS=NITE,LINES=40,ORG=UGAIBM1.LOCAL
// EXEC FORTVCLG,REGION=2000K
//FORT.SYSIN DD *

      read(5,10) cent
   10 format(f6.2)
      fahr=(cent*9/5)+32
      write(6,20) cent,fahr
   20 format(f6.2,' cent = ',f6,2,'fahr')

/*
//GO.SYSIN DD *
100.00
/*
//
```

10.2 EXAMPLE SAS PROGRAM

In this example, the userid RESRCH is executing a SAS procedure with the SAS program inserted after the SYSIN line and the data is being read from the first file of a standard labeled cartridge tape with the file name of DATAFIL. The output will go directly to the printer with the sysout class of S.

```
//ANALYSIS JOB USER=RESRCH,NOTIFY=RESRCH,TIME=3,MSGCLASS=S
//*MAIN LINES=10
//*OPERATOR VRN=#C1234 VSN=WGCF KEY=KFJD
// EXEC SAS,REGION=2500K
//SAMPDATA DD UNIT=TAPECA,VOL=SER=WGCF,
// LABEL=(1,SL,,IN),
// DSN=DATAFIL,
// DISP=(OLD,PASS)
//SYSIN DD *

data one;
   infile sampdata;
```

```
    input a b c d;
proc print;

/*
//
```

11. SOURCES OF HELP

11.1 UCNS HELPDESK

The UCNS Helpdesk, located in the Computer Services Annex on the corner of East Campus Road and Cedar Street and on the first floor of the Boyd Graduate Studies Research Center, function as the initial contact point for all computer related needs associated with the University of Georgia computer systems. The primary purpose of the Helpdesk is to aid faculty, staff, and students with the use of our computer systems through general consulting and information requests. Assistance is provided through telephone contacts, walkins, and electronic mail.

Hours: Monday - Friday, 8:00AM - 5:00PM
Phone: (404) 542-3106

E-mail Address: HELPDESK@UGA

Internet Address: HELPDESK@ARCHES.UGA.EDU

Limited assistance is available at all staffed UCNS Computer Service Sites.

11.2 HELPFUL IDS DOCUMENTS

- Common IBM System Error Messages and Abend Codes
- The Case of the Sinister Syntax Error
- Facility Access and Services Guide

11.3 BASIC JCL REFERENCES

- IBM MVS JCL Publication No. GC28-1300 - This manual is the basic reference document on the syntax and usage of IBM JCL. This manual is available for reference at the Client Services Help Desk.
- System/370 Job Control Language by Gary Deward Brown - This reference is a standard and popular textbook for introducing IBM JCL to persons familiar with computers but not necessarily IBM 370 systems.